


C# Avançado

Reflection e Attributes




Tópicos Abordados



- Reflection
 - A Classe *System.Type*
 - Obtendo objetos *Type*
 - Membros da classe *Type*
 - Exemplos
- Attributes
 - Usando attributes
 - Criando attributes
 - Múltiplos attributes
 - Restringindo o uso de attributes

Reflection e Attributes

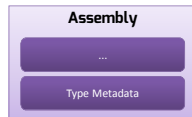


- O uso de reflection e a criação de attributes normalmente não são necessários em aplicações “normais”
- Onde é possível aplicar estes conceitos?
 - Em aplicações voltadas a desenvolvedores
 - Frameworks
 - Ferramentas

Type Metadata



- Todo assembly contém os metadados dos tipos de dados que ele define



- Os metadados armazenam informações detalhadas de cada tipo de dado (classes, estruturas, delegates, etc.)
 - Métodos, fields, construtores, etc.

Reflection




- Reflection (ou reflexão) é o processo de descobrir informações sobre tipos de dados durante a execução da aplicação
- A reflection usa os metadados dos tipos definidos no assembly para obter estas informações

A Classe *System.Type*



- A classe *Type* tem um papel central no processo de reflection
- Um objeto do tipo *Type* representa um tipo de dado
 - Todo tipo de dado definido em .NET tem um objeto *Type* associado, que descreve o tipo

Obtendo o Objeto *Type*



- Através do operador *typeof()*
 - Ideal quando você não tem um objeto

```
Type t = typeof(int);
```

```
Type t = typeof(Pessoa);
```

- Através do método *GetType()*
 - Método definido na classe *System.Object*
 - Ideal quando você tem um objeto


```
int i = 10;
```

```
Type t = i.GetType();
```

```
Pessoa p = new Pessoa();
```

```
Type t = p.GetType();
```


Obtendo o Objeto *Type*



- Através de uma string que representa o tipo
 - Ideal quando você não tem acesso ao tipo durante a fase de compilação, ou não quer criar essa dependência no código

```
Type t = Type.GetType("App.Pessoa");
```

Membros da Classe *Type*



- O objeto *Type* é a porta de entrada para o serviço de reflection
- Type* possui uma série de membros

Membro	Informação obtida	Retorno
<i>GetMethods()</i>	Lista de métodos	<i>MethodInfo[]</i>
<i>GetFields()</i>	Lista de fields	<i>FieldInfo[]</i>
<i>GetProperties()</i>	Lista de properties	<i>PropertyInfo[]</i>
<i>GetConstructors()</i>	Lista de construtores	<i>ConstructorInfo[]</i>
<i>GetInterfaces()</i>	Lista de interfaces	<i>Type[]</i>
<i>IsClass</i>	É uma classe	bool
<i>IsAbstract</i>	É abstrato	bool
<i>IsValueType</i>	É um value type	bool

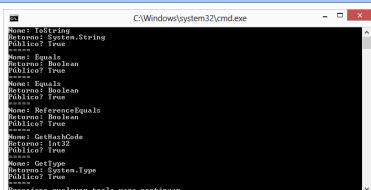
Exemplo do Uso de Reflection



- Mostrar informações sobre métodos da classe *object*

```
Type t = typeof(object);
MethodInfo[] methods = t.GetMethods();

foreach (MethodInfo method in methods)
{
    Console.WriteLine("Nome: " + method.Name);
    Console.WriteLine("Retorno: " + method.ReturnType.FullName);
    Console.WriteLine("Público? " + method.IsPublic);
    Console.WriteLine("=====");
}
```



```
C:\Windows\system32\cmd.exe
Nome: ToString
Retorno: System.String
Público? True
Nome: Equals
Retorno: Boolean
Público? True
Nome: GetHashCode
Retorno: Int32
Público? True
Nome: GetType
Retorno: System.Type
Público? True
Pressione qualquer tecla para continuar. . .
```

Exemplo do Uso de Reflection



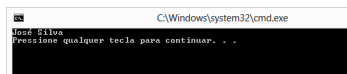
- Definindo um valor para uma property

```
class Pessoa
{
    public string Nome { get; set; }
}
```

```
Pessoa p = new Pessoa();
Type t = p.GetType();

PropertyInfo propInfo = t.GetProperty("Nome");
propInfo.SetValue(p, "José Silva");

Console.WriteLine(p.Nome);
```



```
C:\Windows\system32\cmd.exe
José Silva
Pressione qualquer tecla para continuar. . .
```

Instanciando Objetos com Reflection



- A reflection pode ser usada para instanciar objetos
 - A chamada `Activator.CreateInstance()` é utilizada

```
Type t = typeof(Pessoa);
Pessoa p = (Pessoa) Activator.CreateInstance(t);
```

Attributes



- Attributes (atributos) são metadados extras que você pode fornecer a tipos e elementos (classes, estruturas, métodos, fields, etc.)
- O uso de atributos só tem sentido se algum código fizer algo com eles
 - A leitura dos atributos é feita via reflection

Attributes no Código



- A plataforma .NET possui uma série de atributos que podem ser usados
- Você pode criar seus próprios atributos

```
class Pessoa
{
    [Obsolete]
    public void Dormir() { }
}

static void Main()
{
    Pessoa p = new Pessoa();
    p.Dormir();
    // 'Pessoa.Dormir()' is obsolete
}
```

Atributo

O compilador descobre o atributo via reflection e avisa o programador

Criando Atributos



- Um atributo é uma classe que herda de *System.Attribute*
- Por convenção, o nome da classe deve terminar com *Attribute*
- Recomenda-se que a classe do atributo seja definida como *sealed*
 - Previne a herança

Criando Atributos

```
sealed class AuthorAttribute : Attribute
{
    public string Name { get; set; }
    public AuthorAttribute(string name)
    {
        this.Name = name;
    }
    public AuthorAttribute() { }
}
```

Sufixo Attribute

Herda de Attribute

Property

Construtores

O sufixo pode ser omitido

Define um valor para a property

Invoca o construtor que recebe uma string

```
[Author(Name = "Julio")]
class Pessoa { }
```

```
[Author("Julio")]
class Pessoa { }
```

Buscando Atributos via Reflection

- O processo de reflection deve ser usado para obter informações sobre os atributos

```
Type t = typeof(Pessoa);
AuthorAttribute author = t.GetCustomAttribute<AuthorAttribute>();

if (author != null)
{
    string name = author.Name;
}
```

Múltiplos Atributos

- Elementos podem ter múltiplos atributos caso seja necessário

```
[Author("Julio")]
[Obsolete]
class Pessoa { }
```

```
[Author("Julio"), Obsolete]
class Pessoa { }
```

Restringindo o Uso de Atributos



- Por padrão, um atributo pode ser usado em qualquer elemento (classe, estrutura, método, construtor, field, etc.)
- É possível restringir o seu uso

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Struct)]  
sealed class AuthorAttribute : Attribute { }
```

Limita o uso a classes
ou estruturas

Consulte a documentação do
enum `AttributeTargets` para
conhecer outras possibilidades



Softblue
